Proteus, des web services pour les systèmes de maintenance

X. Rebeuf♦, N. Blanc, F. Charpillet♦, D. Cheve*, A. Dutech♦, C. Lang+, L. Pélissier♦, J.P. Thomesse♦*
* Cegelec, *Nanterre*, France. + LIFC, Besançon, France. ♦ LORIA, Nancy, France
Xavier.Rebeuf@loria.fr

RESUME
Le projet PROTEUS a comme objectif de fournir une plate-forme et les concepts génériques pour construire des systèmes de e-maintenance industrielle incluant les systèmes existants d'acquisition de données, de contrôle – commande, de gestion de la maintenance, d'aide au diagnostic, de gestion de la documentation, etc. Le but de la plate-forme est non seulement d'intégrer des outils existants, mais aussi de prévoir l'évolution de ceux-ci au travers de l'introduction de nouveaux services.
Les concepts de Web services, d'ontologie et de services génériques associés à des modèles génériques des données sont au centre de la solution en cours de développement. En effet, ces techniques permettent de garantir l'interopérabilité de systèmes hétérogènes. Cet article décrit les principes de la solution en s'appuyant sur un exemple de processus et un scénario typique de maintenance corrective. De plus, il décrit les résultats préliminaires obtenus lors des premières expérimentations.

MOTS CLES : e-maintenance, plate-forme d'intégration, Web Services, workflow, serveur de données

1. OBJECTIFS DU PROJET PROTEUS[1]

La maintenance des systèmes industriels est une fonction cruciale pour la qualité des produits et services fournis ainsi que pour une productivité que l'on cherche à améliorer chaque jour. Mais c'est aussi une fonction complexe, tellement complexe que de nombreux systèmes informatiques d'aide, d'assistance, de gestion de certaines fonctions sont devenus nécessaires et doivent cohabiter. Ils sont évidemment plus ou moins complémentaires ou redondants, parfois incohérents, toujours hétérogènes ; ce sont ces raisons qui ont motivé le projet PROTEUS dont l'objectif est de faciliter l'intégration de ces divers systèmes en définissant une description unique et cohérente de l'installation à maintenir (une ontologie), une architecture générique (basée sur le concept de Web Service) et en proposant des modèles et des solutions technologiques d'intégration. Le projet PROTEUS a conduit à définir une plate-forme d'intégration, avec tous les problèmes inhérents à ces technologies, répartition des composants, découverte de services, définition des services, distribution des « workflows », contrôles des accès, etc. Il est hors de question de présenter ici l'ensemble des solutions retenues pour chacun des problèmes. Nous allons nous restreindre aux aspects d'architecture et de workflow qui seront explicités à partir d'un exemple pour concrétiser les concepts généraux et génériques introduits dans ce projet.

Mais avant de rentrer dans cette partie technique, il nous a semblé nécessaire de présenter le contexte de la maintenance industrielle (cf. §2). L'architecture de la plate-forme d'intégration est présentée au paragraphe 3. Un scénario typique de maintenance corrective est introduit au paragraphe 4. Il porte essentiellement sur les interactions entre un système de diagnostic et le système de surveillance du dispositif à maintenir. Ce scénario est décrit volontairement selon une certaine décomposition des rôles des acteurs qui suppose déjà un raffinement du scénario original exprimé par l'utilisateur final. Pour réaliser ce scénario, nous définissons les services génériques des différents outils au paragraphe 5. Le paragraphe 6 illustre comment ces services sont mis en œuvre sur un équipement particulier ; nous y

[1] Cette étude a été effectuée dans le cadre d'un projet Européen PROTEUS financé par le Ministère de l'économie et des finances sous la responsabilité de European Commision Initiative ITEA. Le site Web de ce projet est le suivant http://www.proteus-iteaproject.com.

présentons en fait une instanciation des services et des concepts génériques. Enfin, nous concluons avec les premiers résultats (cf. §7 et 8).

De nombreuses simplifications ont été nécessaires pour respecter un volume limité pour cet article. Elles seront rappelées au cours de cet article, mais nous pensons qu'elles ne nuisent ni à la compréhension ni à l'intérêt des concepts énoncés et décrits.

2. LES COMPOSANTS D'UN SYSTEME DE MAINTENANCE

La complexité de la maintenance a plusieurs causes ; elle est d'abord liée à la complexité croissante des objets, des équipements, des machines, des processus créés par l'homme ; elle est aussi rendue complexe par les difficultés d'accès (off-shore) ou le danger inhérent au procédé (nucléaire, espace…). Mais la complexité est aussi due d'une part au choix des stratégies de maintenance et d'autre part au nombre d'opérations, d'acteurs et de ressources nécessaires.

Les stratégies sont multiples et leur choix n'est ni évident, ni exclusif. Maintenance préventive, prédictive, curative, pour ne citer que celles-ci, cohabitent en général pour améliorer la disponibilité d'un dispositif. Les temps d'arrêt doivent être minimisés partout, qu'il s'agisse de machines de production d'énergie, d'un TGV en panne sur une voie, d'un satellite de télécommunications. Il faudrait même être capable de continuer à fournir le service quel que soit l'état du système!

Le nombre des acteurs et des opérations est en général énorme et leur gestion nécessite des systèmes informatiques d'assistance et de gestion. Le scénario, même simplifié à l'extrême du paragraphe 4, en donne une idée. A partir d'une indication issue en général d'une observation d'un opérateur (voire parfois d'un client), ou d'un système d'acquisition de données, il faut analyser ce qui s'est passé pour identifier l'origine de la défaillance. Cette analyse utilise des modèles, des retours d'expérience, des documents… Puis éventuellement il faut chercher les pièces de rechange, les outils de démontage et remontage, le personnel qualifié, les procédures à mettre en œuvre tant pour la réparation que pour la remise en service, les tests à effectuer, la mise à jour du retour d'expérience… La gestion des ressources évoquées ci-dessus est évidemment une fonction de base (gestion des pièces de rechange, des personnels, gestion financière…).

Pour mener à bien la maintenance d'un système industriel, on utilise en général les sous-systèmes fonctionnels suivants : un système d'acquisition de données (SCADA, Supervisory Control And Data Acquisition), des aides au diagnostic, un système de GMAO (Gestion de Maintenance Assistée par Ordinateur), un ERP (Enterprise Resource Planning), un système de documentation. Chacun de ces systèmes s'appuie sur un certain modèle de l'entreprise, du système physique ou de l'installation à maintenir. Ces modèles sont évidemment différents puisque leurs objectifs le sont, mais ils sont aussi parfois incohérents car définis indépendamment les uns des autres. Les logiciels sont parfois redondants, mais dans tous les cas non interopérables, tant par leurs représentations hétérogènes des informations que par leurs interfaces incompatibles entre elles. Ceci est vrai sauf dans le cas où tout est intégré par construction initiale comme dans [24].

C'est cette recherche d'interopérabilité, et d'intégration harmonieuse des diverses fonctions qui a motivé le projet PROTEUS.

2.1. SCADA

Un SCADA est un système d'acquisition de données qui permet le suivi de la dynamique du système physique, et qui éventuellement détectera les dérives et les alarmes qui nécessiteront une intervention immédiate ou sa programmation à une date ultérieure. Ce système est soit indépendant, soit intégré dans

un système de contrôle - commande, dans des automates programmables, etc. Un SCADA est rarement construit sur un modèle de maintenance. Il modélise le processus physique à des fins de commande et-ou de supervision. Les données pertinentes de maintenance selon la stratégie envisagée ne sont pas forcément stockées, elles peuvent aussi être ni mesurées, ni calculées, et le besoin se fait jour de développer des adaptations en vue d'une intégration. Un cas particulier est celui des instruments intelligents (capteurs ou actionneurs) qui peuvent déjà intégrer quelques fonctionnalités relevant de la problématique de la maintenance comme dans le cas de la plate-forme OntoServ.Net [20],[21].

2.2. Systèmes d'aide au diagnostic

Un ou des systèmes d'aide au diagnostic peuvent être utilisés soit en cas d'arrêt et d'assistance à la recherche de la cause du dysfonctionnement, soit actifs en permanence pour prévenir des défaillances autant que faire se peut. Ces systèmes sont construits sur des modèles très variés (modèles de Markov, de Bayes, modèles neuro-mimétiques, modèles de raisonnement à base de cas…). Ils doivent, en général, s'appuyer sur des données issues d'un SCADA et sur l'expérience acquise pour assister la décision des responsables de la maintenance.

2.3. Système de GMAO

Un système de gestion de maintenance assistée par ordinateur (GMAO) optimise les activités de maintenance et impacte ainsi directement la productivité du matériel maintenu. Il doit gérer les diverses stratégies de maintenance, maintenance corrective, maintenance préventive, prédictive, etc.
Les objectifs d'une GMAO peuvent ainsi être décomposés selon les trois axes suivants :

- gestion proprement dite de l'activité de maintenance,
- génération du retour d'expérience (REX),
- analyse des données tout au long des autres processus.

Grâce à la GMAO, il est alors possible :

- de mieux maîtriser les équipements grâce à une diminution des temps d'arrêt et du nombre de défaillances, une augmentation de la disponibilité des matériels ainsi qu'une optimisation de l'efficacité du personnel,
- de mieux suivre le déroulement des travaux,
- d'optimiser les stocks,
- de réduire les coûts en optimisant les interventions et les stratégies.

2.4. Système de documentation

Un système de gestion de la documentation des équipements est au minimum la base documentaire associée à l'installation à maintenir. Ce système doit délivrer la bonne information, au bon moment, au bon endroit et à la bonne personne et ce tout au long de la procédure. Les informations ont longtemps été textuelles : procédures, relations d'anciennes expériences, notices d'utilisation ou de montage - démontage. Elles sont maintenant multimédia, incluant films et commentaires. Mais les systèmes de documentation contiennent maintenant plus que cela ; ils contiennent tout ce qui est relatif à la qualité, à la sécurité, les normes, les règlements intérieurs, et plus généralement toutes les informations internes à l'entreprise.

2.5. Système ERP

Un système ERP a pour vocation l'optimisation de la productivité de l'entreprise. La méthode est basée sur une gestion, sur une optimisation et surtout une synchronisation des flux de matières, de produits, d'information, de décision et évidemment des flux financiers. De tels systèmes intègrent toutes les informations de l'entreprise, ressources humaines, gestion des achats et de la logistique, service commercial et gestion des ventes, production et gestion des matières, qualité des produits et des services,

gestion comptable et financière, et évidemment la maintenance qui est en relation avec chacune des fonctions précédentes.

3. ARCHITECTURE DE LA PLATE-FORME

Le but de PROTEUS étant d'assurer l'interopérabilité des différents outils présentés précédemment, nous définissons tout d'abord l'architecture de la plate-forme d'intégration (voir Fig. 1) . Une description plus complète est disponible dans [3]. Cette intégration relève d'une démarche plus générale dans l'industrie, comme dans MIMOSA [25], qui est résumée par l'expression "Enterprise Integration Technologies", intégration qui se veut le plus possible dynamique, et pas seulement statique [4], [11], [12].

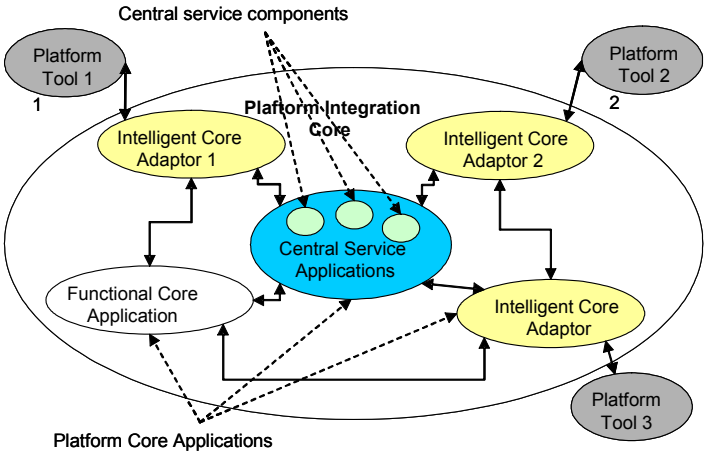

Fig. 1 : Architecture simplifiée de la plate-forme PROTEUS

La plate-forme doit offrir un service global et intégré aux utilisateurs. Pour cela, elle doit résoudre trois types de contrainte :

- la coopération entre des applications de différentes natures (temps réel, transactionnelles, interactives),
- le besoin d'échange d'informations entre sites distants,
- la variété des formats de données.

Les différents outils supportant des fonctionnalités communes offrent rarement la même interface. Pour chaque type de système (ex : SCADA), nous définissons une interface générique standard (voir paragraphe 5). Elle présente ainsi une vision « idéale » de l'outil et permet d'uniformiser l'accès à ces fonctionnalités depuis la plate-forme. L'implémentation de cette interface pour un outil existant est assurée par un type de composant appelé « Intelligent Core Adapter » (voir paragraphe 3.2).

Afin d'assurer l'évolutivité de la plate-forme, il faut pouvoir intégrer de nouvelles fonctionnalités non fournies par un outil pré-existant. Ce type de fonctionnalité sera implémenté au travers d'un type de composants appelé « Functional Core Adapter » (voir paragraphe 3.3).

Enfin, d'autres outils sont nécessaires pour le bon fonctionnement de la plate-forme. Par exemple, la plate-forme nécessite des serveurs d'authentification pour les utilisateurs et d'annuaire des services disponibles. Ces différents outils sont intégrés au sein d'un composant appelé « Central Service Application » (voir paragraphe 3.1). Afin d'assurer l'interopérabilité des échanges, les communications sont basées sur la technologie des « Web Services » et le protocole SOAP (Simple Object Access Protocol) [18], [3], [27].

3.1. Les outils du « Central Service Application »

Le CSA regroupe les différents outils responsables du bon fonctionnement de la plate-forme.

- *Annuaire*. L' « Universal Description, Discovery and Integration » permet l'enregistrement ainsi que la découverte dynamique de Web Services [26]. Cette fonctionnalité est cruciale dans le cas de Proteus. En effet, l'ajout d'un nouvel outil (par exemple un SCADA) sur la plate-forme ne doit pas interrompre son fonctionnement. Il faut donc que les outils existants puissent découvrir les nouveaux services à tout instant.
- *Gestion de sécurité*. Sur la plate-forme, différents acteurs peuvent intervenir. Il faut donc pouvoir gérer les droits des utilisateurs comme des applications (définies par contrat entre la société de maintenance et le client). De plus, cette base d'authentification permet de limiter les accès non autorisés.
- *Base de liens entre les instances.* Chaque outil possédant sa propre ontologie, il est indispensable de créer une base de liens permettant d'associer un équipement (par exemple un moteur décrit dans l'ontologie du SCADA) à sa documentation (décrit dans l'ontologie du serveur de documentation). Cela implique de nommer les objets de manière unique et de permettre une navigation entre les différentes relations définies par les ontologies.
- *Gestion des événements.* Il s'agit de centraliser les abonnements à des évènements (alarmes, valeurs périodiques…). Un tel serveur n'est pas obligatoire. En effet, ces abonnements peuvent être directement souscrits auprès de chaque producteur (par exemple, un SCADA). Le but du serveur d'abonnements est de diminuer le trafic sur la plate-forme.

Notons que pour des raisons de robustesse et de rapidité, ces serveurs peuvent être distribués et/ou répliqués sur les différents sites. Nous ne traiterons pas de ces aspects et de leurs conséquences ici.

3.2. L'« Intelligent Core Adapter »

La plate-forme est basée sur une standardisation des communications. Les interactions doivent donc suivre des schémas de flot de données et de flot de contrôle. Afin que les services offerts par chaque application correspondent à ces schémas, nous introduisons le composant ICA (voir Fig. 2).

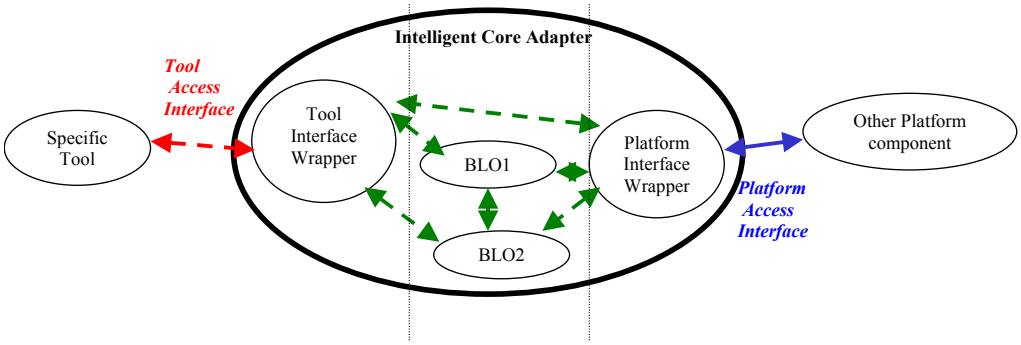

Fig. 2 : Description d'un "Intelligent Core Adapter"

Pour chaque type d'application, nous définissons un ensemble de Web Services standard. Le rôle de l'ICA est donc d'implémenter ces Web Services. Il est constitué de trois parties:

- Une API réalisant l'interface avec l'application existante (« Tool Interface Wrapper »).
- Un ensemble de « Business Logic Objects » : ces objets sont chargés de transformer les données et d'enchaîner les appels de services de l'application afin de réaliser le Web Service. Notons que ces objets peuvent éventuellement faire appel à d'autres Web Services de la plate-forme pour remplir leur mission.

- Enfin, une API réalisant l'interface entre les BLOs et la plate-forme (« Platform Interface Wrapper »). Cette API implémente l'interface de l'outil « idéal » défini par l'ensemble de Web Services standards correspondants.

3.3. Le « Functional Core Adapter »

Il est possible que des services ne correspondant à aucune application pré-existante soient ajoutés à la plate-forme. Pour cela, nous introduisons le FCA. Il reprend la structure d'un ICA à l'exception de la partie API vers une application existante (« Tool Interface Wrapper »).

3.4. Implémentation des scénarii de maintenance

Un des intérêts de la plate-forme PROTEUS réside dans la possibilité d'automatiser tout ou partie des scénarii de maintenance. Chaque étape correspond à l'invocation d'un ou plusieurs services. Les résultats issus d'une étape conditionnent le choix de l'étape suivante. Cet ensemble de scénarii peut donc être vu comme un workflow [1] définissant un service composé. Il devient alors possible d'automatiser le traitement d'une alarme (par exemple) conformément aux scénarii de maintenance préalablement définie (voir paragraphe 4). Bien entendu, certaines étapes nécessitent une confirmation manuelle. Dans ce cas, une des invocations du workflow correspond à un service du portail web.

Afin de transformer un ensemble de scénarii en workflow, certaines étapes s'avèrent nécessaires :

- *Raffiner les services*. Dans les scénarii de maintenance, les différentes étapes spécifiées ne correspondent pas forcément à des services élémentaires des différents outils « idéaux ». Il faut donc décomposer certaines étapes afin de les exprimer en fonction de ces fonctionnalités basiques.
- *Adapter/Transformer les données*. Le format des résultats d'une étape peut ne pas correspondre avec celui admis dans l'étape suivante. Il faut donc ajouter des mécanismes d'adaptation des données.
- *Identifier finement les dépendances*. Afin d'optimiser l'exécution du workflow, certaines étapes de traitement ou d'invocation de services peuvent avoir lieu en parallèle. Il faut donc identifier les différentes dépendances.
- *Assurer la cohérence des données*. En cas de problème lors de l'exécution d'un service, il faut identifier les nouveaux services à invoquer (pour annuler certains ordres par exemple) afin d'assurer la cohérence des informations distribuées sur la plate-forme.
- *Limiter l'automatisation des enchaînements*. Pour des raisons de sécurité ou de faisabilité, certaines décisions d'enchaînement doivent rester à la charge des acteurs humains. En accord avec l'expert en maintenance, il faut donc identifier ces étapes.
- *Placer des points de validation*. Au bout d'un certain nombre d'enchaînements automatiques, il faut placer des points d'arrêt permettant à un acteur humain de valider ou non les étapes correspondantes.

Les enchaînements d'un workflow peuvent être gérés par une ou plusieurs applications. L'exécution centralisée (i.e. sur une seule application) génère beaucoup de trafic en un point de la plate-forme. Nous choisissons donc de distribuer ce workflow sur plusieurs applications (un exemple de distribution sera discuté dans le paragraphe 4).

- Certaines étapes produisent des résultats ne concernant qu'une application particulière. Dans ce cas, elles sont implémentées comme un nouveau « Business Logic Object » d'un ICA existant. Cet ICA offre alors de nouveaux Web Services composés.
- Les grandes étapes sont assurées de façon centralisée. Elles sont implémentées dans un composant « Functional Core Adapter ». Chaque grande étape définit alors un nouveau Web Service du composant.

Les scénarii de maintenance doivent pouvoir évoluer selon les besoins. Il faut donc une implémentation flexible des workflows. Nous proposons d'utiliser la notion de chorégraphie de Web Services [2]. Cette approche présente plusieurs avantages :

- De nombreuses implémentations de chorégraphie proposent un éditeur graphique. Cet aspect est très important car il permet à un spécialiste de la maintenance de valider le codage des scénarii. Une modification de cette dernière peut alors être éditée par cet utilisateur non spécialiste des techniques informatiques.
- Les interpréteurs de chorégraphie fournissent directement des Web Services composés.
- Des formats de codage commun existent et permettent de passer d'un interpréteur à un autre.

Cependant, pour des raisons d'efficacité, il est toujours possible de définir des services composés « figés » (i.e. qui n'évoluent pas) en codant directement l'enchaînement dans un langage de programmation.

4. DU SCENARIO AU WORKFLOW

Dans cette section, nous présentons un scénario simplifié de maintenance et nous en déduisons le workflow et les différents Web Services composés correspondants.

Le scénario de départ est le suivant :

- Une alarme est générée par le SCADA.
- L'utilisateur effectue une demande d'intervention.
- Il déclenche l'outil de diagnostic.
- Ce dernier récupère les données pertinentes auprès du SCADA.
- Il propose un diagnostic.
- L'utilisateur donne son Ordre de Travail et réserve les pièces détachées auprès de la GMAO.
- Il réserve les ressources humaines nécessaires auprès de l'ERP.
- Finalement, il demande à la documentation les gammes correspondantes (i.e. les procédures d'intervention).

La Fig. 3 présente le workflow issu du scénario précédent. Chaque étape a été détaillée de façon à correspondre à un service élémentaire décrit dans le paragraphe 5. Trois Web Services composés et un enchaînement manuel ont été créés.

- *Service Diagnostic.* Il gère toute la partie récupération des données pertinentes du SCADA. Comme ces données ne servent qu'à l'outil de diagnostic, ce service est hébergé dans l'ICA du système expert.
- *Service GestionAlarme*. Il prend en paramètre une alarme du SCADA. Il gère le workflow jusqu'à la présentation du diagnostic à l'utilisateur. Ce service n'étant pas rattaché à un outil existant, nous créons un FCA responsable de ce service.
- *Validation du diagnostic*. Cette étape étant cruciale, elle reste à la charge de l'utilisateur. Si il valide ce diagnostic, alors il invoque le Web Service suivant au travers du portail Web.
- *Service gestionOT*. Il prend en paramètre la demande de l'utilisateur et gère les réservations jusqu'au bout.

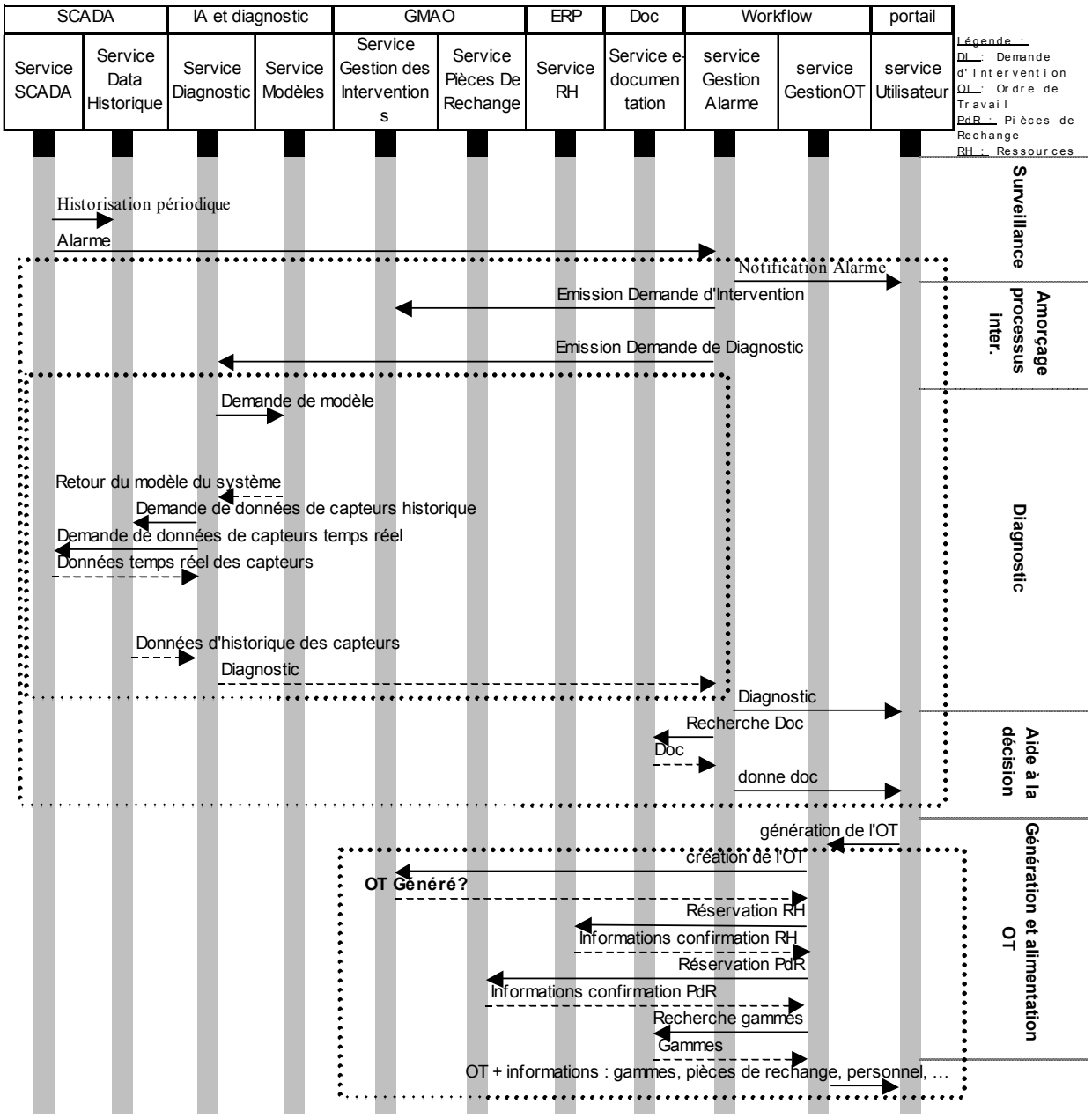

Fig. 3 : Identification d'un workflow partant d'un scénario

Notons que chaque appel à un nouveau Web Service se fait en deux temps :

- Découverte du Web Service grâce au serveur UDDI,
- Appel du service lui-même.

Par manque de place, nous ne décrirons pas la découverte du service. Celle-ci correspond à une découverte « standard » avec cette technologie (par exemple recherche par mot-clef).

5. LES SERVICES GENERIQUES

Dans cette section, nous décrivons les services des différents composants de la plate-forme. Ils sont soit basés sur le modèle client – serveur, soit sur le modèle producteur – consommateur, aussi dénommé « publisher – subscriber ». Selon le modèle Push Publisher, il s'agit d'un service avec la sémantique de « InformationReport » ou de « EventNotification » de MMS [14]. Ce modèle Push Publisher – Subscriber est bien adapté à la transmission d'événements, à condition que les services sous-jacents garantissent que les messages ne peuvent pas être perdus.

Plusieurs SCADA et normes de réseaux industriels proposent des mécanismes similaires par exemple le réseau M-PCCN d'EDF [15] et la norme de télé-contrôle IEC 870-5 [13].

5.1. Les services

Dans cette section, nous décrivons les différents services des composants. Par souci de concision, nous ne définissons ici que les Web Services élémentaires intervenant dans le Web Service composé « Diagnostic ». La définition des autres services suit la même démarche.

Le SCADA. Le serveur de données (i.e. SCADA « idéal ») fournit en fait une image du processus physique et de son éventuel système automatisé à toutes les autres composantes du système de maintenance. Cette image est composée des valeurs des objets pertinents pour la maintenance, et représentatifs de l'état du système maintenu, de ses évolutions temporelles, et des événements survenus.

Les données échangées correspondent à la classe CVQT (Classe Valeur, Qualité, Temps) constituant l'image du processus, l'état des équipements, et l'évolution de la dynamique du système. L'historique des valeurs correspond alors à une collection d'objets CVQT. Un objet de la classe CVQT a les attributs suivants :

- *Nom* est une chaîne de caractères,
- *Valeur* est soit un entier, un réel ou un booléen.
- *Qualité* est une collection d'attributs qualifiant la valeur. Plusieurs types d'attributs peuvent être introduits :
    - concernant la chaîne d'acquisition de données, par exemple un convertisseur analogique numérique n'a pas été saturé, ou le capteur doit être re-calibré.
    - concernant la validité temporelle de la valeur, par exemple la mesure a bien eu lieu dans une fenêtre temporelle donnée, ou la mesure a été correctement synchronisée avec une autre…
- *TimeStamp* correspond à la date de production de la valeur. La date de production peut-être effectivement la date de production à la source dans le cas où le producteur a la capacité de dater un événement, mais ce peut-être aussi à la réception dans le cas contraire.
- *Unité* doit normalement appartenir au répertoire international des systèmes des poids et mesures.
- *Seuil* représente par exemple une limite à ne pas dépasser. *Seuil* est en fait une valeur et l'unité de la valeur mesurée. On distingue habituellement les seuils bas et haut, voire des seuils bas-bas et haut-haut.

Notons que la description des objets échangés ne suffit pas pour offrir les services. Il faut pouvoir aussi nommer de manière non ambiguë un élément du procédé afin d'en demander sa valeur (ex : nommer un capteur). Pour cela, chaque procédé doit être modélisé et une base de liens permet de servir une vision cohérente des instances à la plate-forme (voir exemple section 6).

Les services sont regroupés dans le tableau suivant sans détailler.

Groupe	Service	Description
Accès	GetAttribute	Service pour l'obtention de la valeur d'un attribut
	SetAttribute	Service pour la modification de la valeur d'un attribut
	ReadValue	Service pour l'obtention de la valeur d'une variable (V, Q, T)
	GetAttributeList	Service pour l'obtention de la valeur d'une liste d'attributs
	SetAttributeList	Service pour la modification des valeurs d'une liste d'attributs
	ReadValueList	Service pour l'obtention de la valeur d'une liste de variables (V, Q, T)
	Archive	Service pour la sauvegarde d'une image du procédé dans la base de données
	History	Service pour la recherche d'une image du procédé dans la base de données
	Snapshot	Service pour l'acquisition et la sauvegarde d'une image du procédé.
Gestion	Create	Service pour la création d'un objet par instanciation
	Delete	Service pour la destruction d'un objet
Evén.	EventNotif.	Service pour notifier un événement à un abonné (qui peut être un distributeur d'événements)
	EventNotif-Ack	Service pour la notification avec demande d'accusé de réception
	Acknowledge	Service d'acquittement d'une notification
Rech.	FindTag	Recherche pour localiser un instrument

L'outil de diagnostic. Le diagnostic a pour objectif de déterminer l'état d'un équipement ou d'un procédé à partir d'observations. Il s'agit d'évaluer si le fonctionnement est correct, dégradé, défaillant et de déterminer les composants qui sont en panne ou qui nécessitent une action de maintenance. De nombreux travaux de recherche ont tenté de répondre à cette problématique tant en automatique qu'en intelligence artificielle. L'article [6] établit un excellent état de l'art des différentes méthodes de modélisation actuellement disponibles en intelligence artificielle.

Au sein de Proteus, l'architecture a été conçue afin que plusieurs outils d'intelligence artificielle puissent cohabiter, voire coopérer. Ces outils sont principalement conçus pour remplir des fonctions de diagnostic et de pronostic appliqués à la maintenance (maintenance corrective dans le cas du diagnostic, préventive dans le cas du pronostic). Parmi les outils envisagés, nous pouvons citer les systèmes à base de règles, les réseaux de neurones, les outils de raisonnement à partir de cas, les modèles de Markov cachés et processus décisionnels de Markov, les réseaux Bayésiens. Cette liste n'est bien sûr pas exhaustive et devrait s'enrichir au fur à mesure des besoins.

La mise en place d'un service de diagnostic requiert d'un point de vue méthodologique de remplir différentes étapes : tout d'abord il faut choisir le formalisme le plus adapté au modèle de diagnostic que l'on cherche à mettre en place, ensuite il faut définir ce modèle. Deux possibilités s'offrent au concepteur : l'utilisation de technique d'apprentissage lorsqu'il dispose de bases de données complètes sur le fonctionnement du système, le recueil d'expertise auprès de spécialistes de l'installation pour laquelle on cherche à définir un modèle de diagnostic.

Par exemple, suite à une alarme portant sur un équipement donné on choisira l'ensemble des règles associées à cet équipement, l'historique des observations faites sur cet équipement et on lancera le moteur d'inférence « Pertinence Rule Maker » en chaînage avant pour en déduire le composant défaillant (le chaînage avant est l'un des mécanismes d'inférence des systèmes à base de règles).

Les services offerts par l'outil d'aide au diagnostic sont les suivants :

- *getDiagnostic* prend en paramètre une alarme et une référence de procédé. Elle retourne un diagnostic sous forme d'un document XML donnant le chemin le plus probable dans le modèle.

- *getModel* prend en paramètre une référence de procédé et retourne un document XML définissant le modèle de dysfonctionnement du procédé.

6. EXEMPLE

Pour illustrer notre approche, nous avons choisi d'utiliser un exemple « classique » de la littérature (voir [5]). Cet exemple met en jeu un processus académique qui est néanmoins représentatif des problèmes que l'on peut trouver sur des processus industriels mécaniques ou chimiques. Dans ce paragraphe, nous nous focaliserons sur la définition des modèles nécessaires pour la configuration de la plate-forme.

Notre processus, schématisé à la Fig. 4, est essentiellement constitué de trois réservoirs R_1, R_2 et R_3 reliés entre eux par des tuyaux. Deux arrivées de liquide remplissent les réservoirs R_1 et R_3 avec des débits respectifs de q^i_1 et q^i_3. Sur les tuyaux reliant R_1 et R_3 à R_2, on a placé des vannes V_1 et V_3 qui régulent les débits q_1 et q_3 entre ces réservoirs. Enfin, une dernière vanne V_2 permet au liquide de s'échapper avec un débit q_2. Divers problèmes de maintenance peuvent se poser : corps étranger dans un réservoir, vanne défectueuse, fuite…

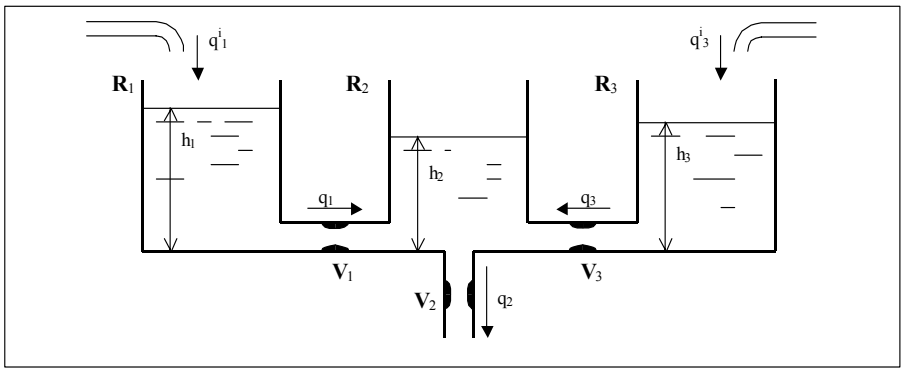

Fig. 4 : Un processus simple mais représentatif.

Pour pouvoir utiliser la plate-forme sur cet exemple, il faut tout d'abord la configurer. Cela consiste en particulier à définir plusieurs modèles :

- le modèle du procédé, permettant de définir la base de liens offrant une vue unifiée du procédé.
- le modèle de dysfonctionnement pour le diagnostic.

6.1. Le modèle du procédé

La Fig. 5 représente le diagramme de classe spécifiant les concepts du procédé. Ce diagramme est essentiel car il permet de définir les relations de navigabilité entre les classes. Ces relations vont servir de modèle aux liens entre les objets dans la base :

- Tout objet instance d'une classe de ce diagramme doit être enregistré dans la base de liens. Cette base lui attribue un identifiant unique.
- Les relations entre les différents objets seront stockées dans la base. Ainsi tous les composants auront une vue cohérente du procédé.

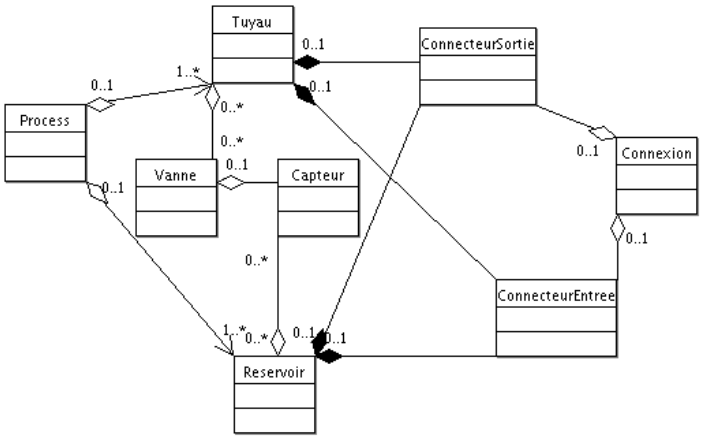

Fig. 5 : Diagramme de classes du procédé

On obtient donc la base d'instances présentée sur la Fig. 6. On y retrouve les trois réservoirs, les trois tuyaux ainsi que les capteurs et les vannes. Cette représentation de la base est simplifiée. En effet, les objets et les relations sont typés conformément au diagramme de classe. Grâce à cette base, il est possible de « naviguer » entre les différents objets. Cette base de données des instances correspond au dernier niveau d'une ontologie décrivant les équipements et les données permettant le diagnostic. Cette approche correspond à celle choisie dans [17].

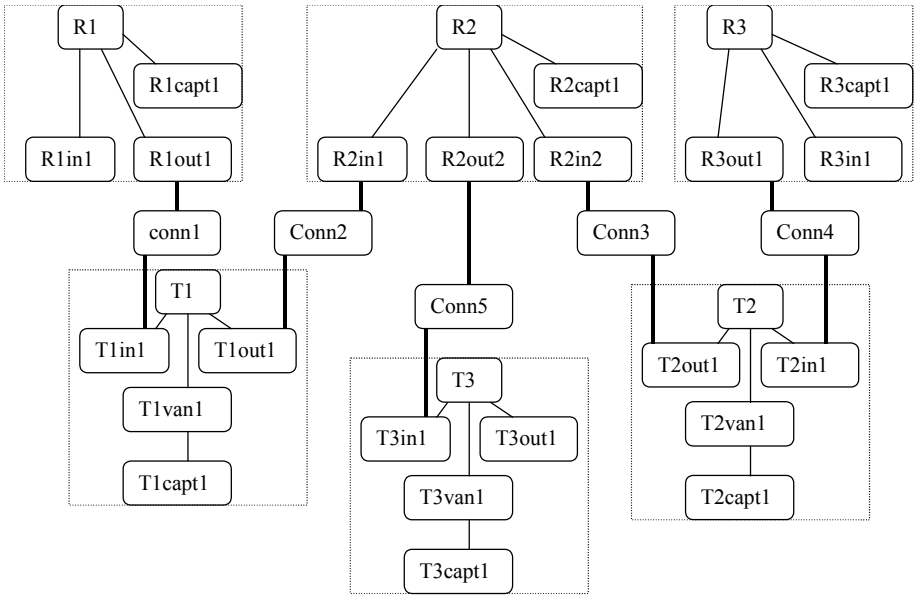

Fig. 6 : Base de donnée des instances

6.2. Le modèle de dysfonctionnement

Le choix du modèle. Dans cet exemple, nous considérons que suite à la demande de diagnostic du service « GestionAlarme », le service « Diagnostic » effectue dans un premier temps une requête vers le service « Modèle ». En fonction du contexte de l'alarme initiale, ce service sélectionne parmi une bibliothèque, le modèle le plus approprié, par exemple un des modèles de l'outil ToolBox [16]. Nous supposons ici que le modèle choisi est exprimé dans le formalisme des processus décisionnels de Markov. Sans entrer dans les détails, disons qu'il s'agit d'un modèle de défaillance qui se représente sous forme d'un graphe probabiliste comme l'illustre la Fig. 7.

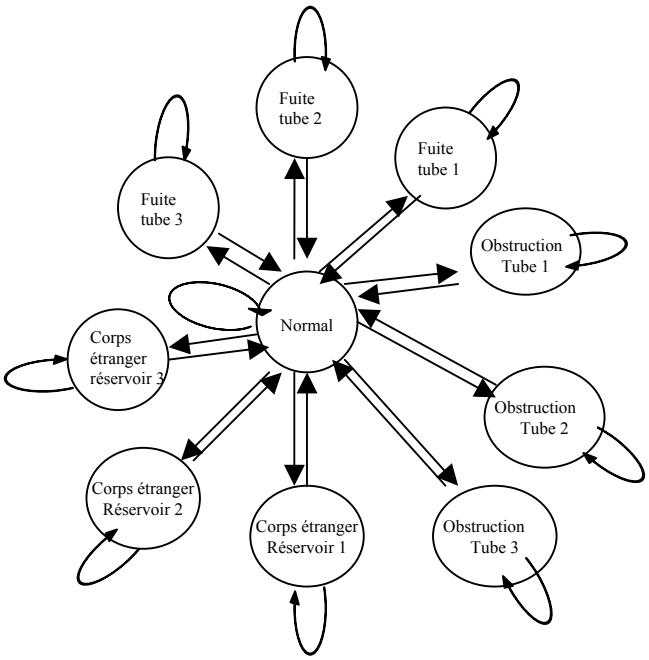

Fig. 7 : Modèle de dysfonctionnement

Cet automate exprime que le processus que nous considérons peut se trouver dans l'un des états de dysfonctionnement suivants :

- Un des trois tuyaux ou un des trois réservoirs fuit
- Un des trois tuyaux est obstrué
- Un corps étranger est tombé dans l'un des trois réservoirs
- Le processus fonctionne correctement

On suppose ici pour des raisons de simplicité, que le processus ne peut se trouver que dans un seul état de dysfonctionnement à la fois et que les débits en entrée des réservoirs 1 et 3 sont constants. Le processus peut passer d'un état à un autre selon une certaine loi de probabilité qui dépend éventuellement des actions de maintenance effectuées sur le processus. Par exemple la probabilité de revenir de l'état « fuite tube 1 » à l'état « normal » lorsque une action de maintenance a été effectuée est proche de 1.

Par ailleurs, chaque état est caractérisé par une fonction d'observation probabiliste qui spécifie ce qu'il est probable d'observer lorsque le processus se trouve dans un état donné. Cela suppose que l'on puisse avoir accès à la hauteur de liquide dans chacun des trois réservoirs et aux débits qui les alimentent. Pour être plus précis, la fonction d'observation du modèle résulte d'un pré-traitement qui vise à rendre ces observations qualitatives. Nous n'entrerons pas dans les détails dans cet article, mais disons simplement que les observations portent plutôt sur les variations des paramètres considérés (par exemple augmentation du débit, augmentation du niveau de la réservoir 1, niveau stable du réservoir 1, ….). Ainsi par exemple, si un corps étranger est tombé dans le réservoir la probabilité d'observer une variation de la hauteur de liquide du réservoir correspondant est importante.

La production du diagnostic. Pour produire un diagnostic, le modèle que nous venons de décrire doit être alimenté par une séquence d'observations sur le processus réel, incluant un historique de quelques minutes. Pour cela le service « Diagnostic » lance deux requêtes vers les services SCADA et « Data Historique » afin de récupérer les dernières valeurs des hauteurs de liquide et derniers débits. Le

diagnostic est obtenu en cherchant la meilleure trajectoire dans l'espace d'état que représente l'automate probabiliste décrit ci-dessus. Cette trajectoire probabiliste, indique à chaque instant de l'historique considéré, la probabilité que le processus soit dans l'un des 10 états mentionnés précédemment. Ce diagnostic est fourni au service gestion d'alarme qui en était demandeur.

Cet exemple illustre comment on peut composer les services afin d'en fournir de plus sophistiqués ou porteur de plus de sémantique. Le problème de la composition (ou de la décomposition) est double, depuis la spécification des services requis par l'utilisateur, jusqu'à la composition semi-automatique [19], voire automatique, en s'appuyant sur la définition de la sémantique des services, mais c'est un autre travail.

7. PREMIERS RESULTATS

Bien que le projet ne soit pas terminé, des résultats préliminaires ont pu être identifiés. Une première partie concerne la définition de la plate-forme elle-même. Une deuxième partie est déduite d'une expérimentation.

7.1. Définition de la plate-forme

Définition des services de base. Pour chaque outil « idéal », nous définissons l'ensemble des services offerts. De plus, ce dernier pouvant être client d'un autre, nous définissons aussi l'ensemble des services requis. Afin d'assurer la cohérence de l'ensemble, nous procédons de manière incrémentale jusqu'à ce que l'ensemble de tous les services requis soit couvert par un ou plusieurs services offerts.

Définition des workflows. Notre but est de déduire les workflows des scénarii de maintenance. Dans un premier temps, nous formons les spécialistes de maintenance à l'outil graphique JAWE permettant d'exprimer directement des workflows.

7.2. Expérimentation

Une première expérimentation de ce modèle est en cours. A cette étape, le but est de valider l'architecture. Donc, nous n'avons pas développé des ICAs pour des outils existants. Ils sont connectés à des simulateurs jouant le rôle d'outil réel.

SOAP. La plate-forme est développée en utilisant deux techniques différentes. La première est basée sur le langage Java. Elle utilise le serveur Apache Tomcat avec Axis. La deuxième repose sur le langage C# et la plate-forme « .net ». Lors de tests de communication, il est apparu que certains types de base ne sont pas correctement transmis et traduits dans les deux langages. Nous définissons donc un sous ensemble de type de base supporté dans Proteus.

UDDI. Nous avons choisi d'utiliser le serveur WSDK d'IBM. L'enregistrement et la recherche au travers de classes ne posent pas de problème. En revanche, la difficulté réside dans la définition d'une taxonomie pertinente pour Proteus. Celle-ci doit pouvoir se déduire de l'ontologie.

Workflow. La machine utilisée pour interpréter le workflow est Shark [23]. Elle permet d'exécuter des workflows définis avec l'éditeur JAWE [23]. L'interfaçage entre le workflow et l'invocation de web service nécessite du développement. Toutefois, les classes développées peuvent être génériques.

De nombreux travaux sont encore en cours. En particulier, nous pouvons citer la mise en place de la base de liens qui instancie l'ontologie. De plus, nous travaillons au développement d'un configurateur pour la plate-forme. En effet, celle-ci est basée sur des solutions génériques. Afin de la déployer, il est nécessaire de disposer d'un outil.

8. CONCLUSION

Un système générique de maintenance devrait en toute rigueur pouvoir être instancié pour servir n'importe quel processus physique selon n'importe quelle stratégie de maintenance, avec n'importe quel système de documentation, etc. Le chemin est encore long avant de disposer d'un tel système. Le projet PROTEUS apporte quelques pierres à ce vaste problème. Ses principaux apports concernent aujourd'hui :

- la possibilité de rendre inter-opérables des sous-systèmes hétérogènes,
- les définitions des ICA qui pour chacun des sous-systèmes conduiront à terme à une certaine normalisation des services des divers constituants
- la description générique des équipements à maintenir en plusieurs niveaux d'abstraction,
- la description générique des données caractéristiques des équipements,
- la transcription systématique des scénarios de maintenance en workflow, ce qui permet une grande liberté aux utilisateurs en leur permettant de faire abstraction des techniques.

Ces solutions sont en cours d'implantation, et sont appliquées à la maintenance de deux prototypes, une machine de fabrication d'ailettes de turbine, et une locomotive diesel électrique. Il reste plusieurs problèmes, de diverses natures, à résoudre de façon générique. En ne considérant que le contexte de ce colloque, on peut citer :

- Les problèmes des erreurs dans les workflows : elles sont aujourd'hui prises en charge par l'utilisateur final, certaines d'entre elles pourraient être traitées automatiquement.
- La gestion du parallélisme et des relations d'ordre (total ou partiel) : dans les workflows on devrait pouvoir gérer des demandes de service parallèles, avec les problèmes de cohérence et les techniques de consensus et d'engagement,
- L'introduction d'autres types de coopération entre les acteurs : on a identifié le besoin de coopérations autres, comme des modèles de type client multiserveur déjà introduits au-dessus des éléments de service application comme MMS [7].
- La gestion de la qualité de service : chaque composant devrait être qualifié, les demandes devraient l'être aussi et la recherche des services devrait tenir compte de la qualité requise.

Les technologies actuelles comme UDDI, WSDL ou SOAP devraient être enrichies pour permettre l'automatisation de certaines opérations comme la composition de Web services et la construction de workflows, les concepts du Web sémantique pourrait apporter certaines facilités [8]. Le domaine de la maintenance est certainement l'un des plus directement concerné en particulier pour la prise en compte dynamique et en temps réel de la capitalisation des retours d'expérience dans une base de connaissances associée à l'ontologie. De telles solutions apporteraient non seulement une simplification de la gestion des stratégies de maintenance pour les utilisateurs, mais aussi une optimisation des ressources.

9. BIBLIOGRAPHIE

[1] W. van der Aalst and K. van Hee. *Workflow Managment: Methods, Models, and Systems.* MIT Press, 2002

[2] A. Arkin et al. *Web Service Choreography Interface 1.0.* W3C Note 2002. http://www.w3.org/TR/wsci/

[3] T. Bangemann, E. Garcia, C. Lang, X. Rebeuf, J. Szymanski, J-P. Thomesse and M. Thron. *PROTEUS – a European initiative for e-maintenance platform development.* In IEEE International Conference on Emerging Technologies and Factory Automation, Lisbon, Portugal, September 2003.

[4] C. Bussler, D. Fensel, N. Sadeh. *Semantic Web Services and their role in Enterprise Application Integration and E-Commerce.* March 2003. http://www.gvsu.edu/ssb/ijec/announcements/semantic.doc

[5] C. Close and D. Frederick. *Modelling and analysis of dynamics systems*. John Wiley & Sons, Inc. 1995.
[6] M.O. Cordier, I. Grosclaude, R. Quiniou et S. Robin. *Étude du pronostic pour la maintenance conditionnelle.*IRISA , 2002. Rapport de fin de contrat EDF 101C0002.
[7] Y. Dakroury et J.P. Elloy. *A new multi-server concept for the MMS Environnement.* Proc of 9th IFAC Workshop on DCCS, 1989.
[8] D. Fensel and C. Bussler. *The Web Service Modeling Framework WSMF*. in Electronic Commerce Research and Applications, Vol 1, Issue 2, pp.113-137, 2002.
[9] J. Heflin (Ed.). *Web Ontology Language (OWL) Use Cases and Requirements.* W3C Working Draft 31 March 2003, 2003.
[10] IEC 61158. *Fieldbus standard, profile* 1-2002.
[11] IEC 62264-1. *Enterprise control system integration- part 1 : Models and terminology*. TC 65A, 2003
[12] IEC 62264-2. *Enterprise control system integration- part 2: object models attributes.* TC65A / 408/FDIS
[13] IEC 870-5. *Telecontrol equipment and systems*. 1990.
[14] ISO 9506. *Manufacturing Message Specification*.1998.
[15] O. Jaray, J.-P. Tavella, J.-P. Thomesse. *A messaging protocol for the EDF - PCCN project*. CIRED June 1999, Nice.
[16] L. Jeanpierre. *Apprentissage et adaptation pour la modélisation stochastique de systèmes dynamiques réels.* Thèse à l'Université Henri Poincaré, Nancy 1, décembre 2002.
[17] O. Kaykova, O. Kononenko, V. Terziyan and A. Zharko. *Community formation scenarios in peer-to-peer web service environments.* IASTED, Int. Conf. on Databases and Applications, 17-19 February 2004, Innsbrück, Austria.
[18] N. Mitra (ed). *SOAP, Version 1.2*. W3C recommandation, 7 may, 2003. http://www.w3.org/TR/soap12-part0/
[19] E. Sirin, J. Hendler and B. Parsia. *Semi-automatic composition of Web services using semantic description.* Workshop ICEIS 2003.
[20] V. Terziyan. *Semantic web services for smart devices based on mobile agents.* 2003. vagan@it.jyu.fi
[21] V. Terziyan and O. Kononenko. *Semantic Web Enabled Web Services : State-of-the art and industrial challenges*. ICWS-2003, International Conference on Web Services, June 23-26 2003, Las Vegas. USA
[22] J.P. Thomesse and P. Noury. *Communication models Client - Server vs Producer – Distributor - Consumer.* Contribution to ISO TC 184/SC5/WG2-TCCA, 1989.
[23] http://www.objectweb.org/
[24] http://www.predict.fr/
[25] http://www.mimosa.org/
[26] http://www.uddi.org/
[27] http://www.w3.org/